REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		FINAL/01 FEB 92 TO 30 JUN 95

4. TITLE AND SUBTITLE
ORTHOGONAL INTERCONNECTION NETWORKS FOR MASSIVELY PARALLEL COMPUTERS

5. FUNDING NUMBERS
2304/FS
F49620-92-J-0126

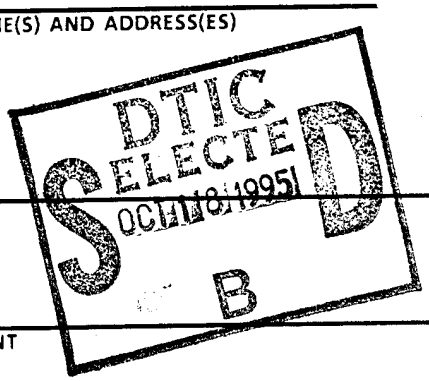
6. AUTHOR(S)
PROFESSOR ISAAC D. SCHERSON

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEPARTMENT OF INFORMATION & COMPUTER SCIENCES
UNIVERSITY OF CALIFORNIA
IRVINE, CA 92717-3425

8. PERFORMING ORGANIZATION
AFOSR-TR-95
0652

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
110 DUNCAN AVE, SUTE B115
BOLLING AFB DC 20332-0001

AGENCY REPORT NO.
F49620-92-J-0126



11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This final report is a cumulative summary of our group's research accomplishments under the AFOSR grant number F49620-92-J-0126. Over the past three years, our work has spanned the areas of architectures, operating system policies, performance analysis and algorithms for parallel processing.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR(SAME AS REPORT)

AFOSR Final Technical Report

Grant Number: F49620-92-J-0126

Proposal: Orthogonal Interconnection Networks for Massively Parallel Computers (Original Control Number 92NM002)

Program Director: Major David Luginbuhl, AFOSR/NM.

Principal Investigator: Professor Isaac D. Scherson.

Research Associates: Brian D. Alleyne[1], Luis Miguel Campos, Chi-Kai Chien[2], David A. Kramer[3], Umesh Krishnaswamy, Vara Ramakrishnan, Veronica L. Reis, Raghu Subramanian.

Department of Information & Computer Science
University of California, Irvine, CA 92717-3425

[1]Currently with Integrated Telecom, Washington, D.C.

[2]Currently with Factset Corporation, San Mateo, CA.

[3]Currently with Thinking Machines Corporation, Cambridge, MA.

Contents

1 Introduction 4

2 Research Objectives 4

3 Research Accomplishments 5

4 Papers Acknowledging the Grant 6

5 Personnel Supported by the Grant 8

6 Significant Events 9

7 Research Interactions 9

7.1 Advisory Consultation 9

7.2 Distinguished Lectures by P.I. 10

7.3 Other Professional Activities of P.I. 11

A Data-Routing Networks 12

B Barrier-Synchronization Networks 13

C Performance Evaluation 13

D Routing Algorithms 15

D.1 On-line Routing 16

D.2 Off-line Routing 17

E Dynamic Precision Algorithms 18

F Parallel Job Scheduling 19

G Parallel Virtual Memory 20

1 Introduction

This final technical report is a cumulative summary of our group's research accomplishments under the AFOSR grant number F49620-92-J-0126. Over the past three years, our work has spanned the areas of architectures, operating system policies, performance analysis and algorithms for parallel processing.

In Section 2, we describe what the objectives of this research were. In Section 3, we outline our research accomplishments and progress towards achieving the above objectives. Section 4 gives a chronological list of papers acknowledging this grant. Section 5 lists the graduate students and principal investigator suppported by this grant. In Section 6, we mention significant events like promotions and graduations. Finally, professional interactions (such as advisory consultations and invited talks) are covered in Section 7. (There are also a set of appendices with detailed information on our research, and a set of graphical slides to complement the appendices.)

2 Research Objectives

Our original proposal (in 1992) raised several problems related to interconnection networks.

As we went along, it became increasingly clear that various architectures and OS issues depend on each other. Problems in one area quickly led to problems in another. Between 1992 and now, the scope of our work expanded from interconnection networks to span such areas as synchronization, performance evaluation, job scheduling and virtual memory.

One of the larger goals of our research was to demonstrate that there *is* a middle-ground between formal but inapplicable research (which the academia is often accused of), and practical but heuristic work (which comes out of the industry). This report describes our fruitful collaboration with MasPar Corporation as one example where we successfully found this middle-ground.

3 Research Accomplishments

This section describes our research accomplishments under the grant. We must restrict ourselves to a quick summary here; for more details, the reader is referred to the appendices, as specified by the table below:

Area	*Research Topic*	*Appendix*
Architecture	Design of Data-Routing Network	A
	Design of Barrier-Synch Network	B
	Performance Evaluation	C
Algorithms	Routing Algorithms	D
	DP (Numerical) Algorithms	E
Operating Systems	Parallel Job Scheduling	F
	Parallel Virtual Memory	G

Design of Data-Routing Network: Most commercial massively parallel computers such as Cray T3D, Thinking Machines CM-5, use bi-directional networks. However there is little insight into *why* they chose the network that they chose. Chi-Kai Chien's thesis developed a comprehensive methodology to answer the above question.

Design of Barrier-Synchronization Network: Synchronization barriers are often nested within one another due to the control block structure. Past methods implement such nested barriers in software. Vara Ramakrishnan has, on the other hand, developed two *hardware* schemes to implement nested barriers — these have the advantage of being more than an order of magnitude faster than the software schemes.

Performance Evaluation: We proposed a new approach to performance evaluation based on performance vectors. This approach enables the estimation of detailed machine characteristics, rather than naïveMflop ratings, and consequently yields many valuable insights into bottlenecks *etc.*. Umesh Krishnawamy's thesis demonstrates the success of this approach on both workstations and supercomputers.

Routing Algorithms: We developed both on-line and off-line algorithms for a class of networks called Expanded Delta Networks (EDN). Since the global router on the Mas-

Par MP-1 and MP-2 is similar to an EDN, we were able to adapt all of our algorithms to it. It turned out that our routing algorithms were significantly better than the ones used by MasPar itself; this has resulted in a fruitful collaboration and technology-transfer. (A letter from MasPar's Director of Architecture is included as Enclosure 1.) This work was by Brian Alleyne and Raghu Subramanian.

Dynamic Precision Algorithms: Dynamic Precision (DP) numerical algorithms vary the precision of data during program execution in order to improve performance. We proposed an architecture called P^3, whose multigauging abilities make it particularly suited to DP algorithms. David Kramer's thesis showed that, by carefully reducing the precision of data when safe to do so, one can achieve a 3x speed-up on the average.

Parallel Job Scheduling: Parallel job scheduling is the problem of how to share a parallel machine among several parallel jobs. There are two orthogonal choices — time-slicing and space-slicing. Raghu Subramanian's thesis proves that, contrary to popular wisdom, time slicing is the most efficient scheduling policy when jobs arrive and depart from the system dynamically.

Parallel Virtual Memory: Although well understood in traditional operating systems, virtual memory is not a reality for parallel computers. Our research attempts to understand the parameters that influence the efficacy of virtual memory. Ongoing experiments point in the direction of dynamic paging policies.

4 Papers Acknowledging the Grant

1. B.D. Alleyne and I.D. Scherson, Expanded Delta Networks for Very Large Parallel Networks, *Proceedings of the International Conference on Parallel Processing,* pp I-127–131, August 1992.

2. C-K. Chien, and I.D. Scherson, Self Routing Least Common Ancestor Networks, *Proceedings of the Frontiers of Massively Parallel Processing,* pp 513–514, October 1992, (Poster paper).

3. B.D. Alleyne and I.D. Scherson, Permutation Routing in 2-Stage Recirculating Delta Networks, *Proceedings of the Frontiers of Massively Parallel Computation,* Vol. 1, pp. 502–503, October 1992, (Poster Paper).

4. D.A. Kramer, Efficient Bit-Parallel Supercomputer Architectures and Algorithms, Ph.D. Thesis, Princeton University, January 1993.

5. [4] I.D. Scherson, and C.-K. Chien, Least Common Ancestor Networks, *Proceedings of the International Parallel Processing Symposium,* pp 507–513, April 1993.

6. [4]I.D. Scherson, and R. Subramanian, Efficient Off-line Routing of Permutations on Restricted Access Expanded Delta Networks, *Proceedings of the International Parallel Processing Symposium,* pp 284–290, April 1993.

7. I.D. Scherson and A.S. Youssef (Editors), *Interconnection Networks for High-Performance Parallel Computers*, IEEE Computer Society Press, Los Alamitos, CA, 1994.

8. B.D. Alleyne and I.D. Scherson, On Evil Twin Networks and the Value of Limited Randomized Routing, *Proceedings of the Proceedings of the International Parallel Processing Symposium,* pages 566-575, April 1994.

9. R. Subramanian and I.D. Scherson, An Analysis of Diffusive Load Balancing, *Proceedings of the 6th ACM Symposium on Parallel Algorithms and Architectures,* pages 220–225, June 1994, Cape May, NJ.

10. B.D. Alleyne, Methodologies for Analysis and Design of Data Routers in Large SIMD Computers, Ph.D. Thesis, Princeton University, June 1994.

11. I.D. Scherson, and C-K. Chien, Least Common Ancestor Networks, *VLSI Design,* Volume 2, Number 4, pages 353–364, April 1995.

12. V. Ramakrishnan, I.D. Scherson and R. Subramanian, Efficient Techniques for Fast Nested Barrier Synchronization, *Proceedings of the 7th ACM Symposium on Parallel Algorithms and Architectures,* pages 157–164, July 1995, Santa Barbara, CA.

[4]Due to a clerical error, AFOSR 90-0144 was acknowledged instead of AFOSR F49620-92-J-0126.

13. F. Chen and V.L.M. Reis and I.D. Scherson, A Study of Parallel Input/Output Subsystems, *Proceedings of the Symposium on Advanced Parallel Processsing Technologies*, September 1995.

14. C-K. Chien, Bi-directional Interconnection Networks for Massively Parallel Computers, Ph.D. Thesis in preparation, University of California at Irvine, September 1995.

15. U. Krishnaswamy, Computer Evaluation Using Performance Vectors, Ph.D. Thesis in preparation, University of California at Irvine, December 1995.

16. R. Subramanian, A Framework for Parallel Job Scheduling, Ph.D. Thesis in preparation, University of California at Irvine, December 1995.

17. R. Subramanian and I.D. Scherson, Networks for Multiple Disjoint Barrier Synchronizations, Under submission to *the International Parallel Processing Symposium,* 1996.

18. V.L.M. Reis and I.D. Scherson, A Virtual Memory Model for Parallel Supercomputers, Under submission to *the International Parallel Processing Symposium*, 1996

5 Personnel Supported by the Grant

Principal investigators:

- Isaac. D. Scherson

Graduate students :

- Chi-Kai Chien — graduated with Ph.D. in September '95 from University of California, Irvine. Thesis title: *Bi-directional Interconnection Networks for Massively Parallel Computers.*
- Umesh Krishnaswamy — expected to graduate with Ph.D. in December '95 from University of California, Irvine. Thesis title: *Computer Evaluation Using Performance Vectors.*
- Raghu Subramanian — expected to graduate with Ph.D. in December '95 from University of California, Irvine. Thesis title: *A Framework for Parallel Job Scheduling.*

- Vara Ramakrishnan
- Shih-Ta Huang
- Kathryn Morse

6 Significant Events

Sep 1995: Chi-Kai Chien awarded Ph.D. (University of California, Irvine). Thesis title: *Bi-directional Interconnection Networks for Massively Parallel Computers.*

Jun 1994: Brian D. Alleyne awarded Ph.D. (Princeton University). Thesis title: *Methodologies for Analysis and Design of Data Routers in Large SIMD Computers.*

1994: P.I. appointed Director, Irvine Research Unit in Advanced Computing, University of California, Irvine.

1994: P.I. appointed Professor, Information & Computer Science and Electrical & Computer Engineering, University of California, Irvine.

Jan 1993: David A. Kramer awarded Ph.D. (Princeton University). Thesis title: *Efficient Bit-Parallel Supercomputer Architectures and Algorithms.*

7 Research Interactions

7.1 Advisory Consultation

We pride ourselves for being able to link our basic theoretical research to industrial problems of practical importance. For example,

- MasPar Computer Corporation has used our results in designing the network architecture (topology and on-line algorithm) of their next-generation massively parallel computer.

- Furthermore, the off-line routing code that we designed for the MasPar network (see **Enclosure 2**) is now used by MasPar engineers for in-house applications, and has been distributed over the Internet to MasPar users in US universities, national laboratories and private corporations to help speedup their scientific applications.

Enclosure 1 is a letter from MasPar attesting to the success and desirability of this technology tranfer.

In addition, in 1993-94, the P.I. served as a consultant to Hughes Research Laboratories on the subject of optical interconnects.

7.2 Distinguished Lectures by P.I.

Universities:

University de Lille, France (Jun '93, Dec '94);
UNAM, Mexico (October '94);
University of Chile (Jul '93);
ETCA–Paris, France (Jun '93);
University of Paris (Jussieu), France (Jun '93);
University of Wroclaw, Poland (May '93).

Conferences:

LATIN'95, Latin American Theoretical INformatics, Chile (April '95);
XIV International Conference of the Chilean Computer Society, Chile (Nov '94);
CIIE'94: 1st Interactive Congress on Electronics, Mexico (Oct '94);
SuperComp 94, Brazil (Sep '94);
IEEE/USP International Workshop on High Performance Computing, Brazil (Mar '94);
International Symposium on High Performance Computing, Chile (Dec '93);
Chilean Workshop on Systems Engineering, Chile (July '93);
Polish-American Symposium on Models for High Speed Computing, Poland (May '93).

Industry:

Thinking Machines Corporation, Cambridge, MA (May '92).

7.3 Other Professional Activities of P.I.

Program Committees:

Chair, XV International Conference of the Chilean Computer Science Society, Arica, Chile (Nov '95)

Member, PDCS '95, Washington, DC (Oct '95);

Member, LATIN'95, Chile (Apr '95);

Member, IPPS '94, Mexico (Apr '94);

Member, IPPS '93, Newport Beach, CA (Apr '93);

Workshop Panels:

Member, IFIP - WG10.3, Concurrent Systems, International Conference on Applications in Parallel and Distributed Computing, Venezuela (Apr '94).

Member, IPPS '92, Los Angeles, CA (Apr '93);

Chair, Frontiers '92, MD (Oct '92)

A Data-Routing Networks

In bi-directional networks, both network inputs and network outputs are associated with the first stage, and messages travel forward to some (possibly intermediate) stage and then backwards, so that some stages may be traversed twice. (See **Enclosure 3**.)

Many prototype and commercial parallel computers (e.g., the University of Texas TRAC 1.1 and 2.0 [18, Sec. 7.3,8.2,8.3] and Thinking Machine Corporation's CM-5 [16]) use bi-directional networks. However there is little insight into *why* they chose the network that they chose. Why was that particular topology and routing algorithm most appropriate to the situation, and no other?

We developed a *methodology* to answer the above kind question. The architect starts with certain externally imposed design constraints such as the number of processors, the length of the messages that will be sent *etc.*He then follows the steps of the methodology, which is a sequence of well-defined analytical and simulations steps. At the end, he will arrive at the best network for the given constraints. This process is illustrated in **Enclosures 4 and 5**.

The idea behind the methodology is to define a very general *design space*, and search this space for the point of least cost that satisfies all given constraints. This idea is depicted in **Enclosure 6**. Each element of design space is a ⟨topology, routing algorthm⟩ pair. We allowed the "topology" component to vary over all Least Common Ancestor Networks. (LCANs are an extremely general class of bi-directional networks, and includes (to the best of our knowledge) every popular bi-directional network in the literature [15, 16, 18, 25] as particular instantiations.) Similarly we let the "routing-algorithm" component vary over circuit-switching and worm-hole routing with various buffer sizes.

Needless to say, such a methodology is a big step forward from *ad hoc* network design. It yields several invaluable insights and design tradeoffs. And it may be used to anwer such non-trivial questions as: How should the routing algorithm change if the number of processors were to increase, or if the technology suddenly allowed more pins per chip? Further details may be found in Chi-Kai Chien's thesis [7].

B Barrier-Synchronization Networks

Data parallel programs involve a form of synchronization called *barrier synchronization*: A point in the code is designated as a barrier and no processor is allowed to cross the barrier until all processors involved in the computation have reached it. Since data parallel programs involve frequent barrier synchronizations, a computer intended to run data parallel programs must implement them efficiently. For this purpose, current MIMD computers, including the CM-5 and T3D, provide a dedicated *barrier tree* exclusively for barrier synchronizations [8, 24].

Current MIMD computers provide just one barrier tree per user application. However, if a data parallel program has control blocks nested within one another, then it requires the simultaneous use of *more than one barrier* synchronization tree, one for each level nesting. (In some special cases, the extra barrier trees can be avoided, but this can not be counted on.)

Since providing as many trees as the number of nestings in programs is not feasible due to cost constraints, current machines solve this problem by also implementing barriers in software (using shared semaphores or complicated message passing protocols). However software barriers are usually an order of magnitude slower than hardware barriers.

We have developed two *hardware schemes* for supporting nested barriers using only limited hardware. The first scheme uses two single-bit-trees to support any number of nested barriers. The method relies on code transformations, and it increases the code size. The second scheme uses an integer-tree, which requires more expensive hardware, to support an exponential number of nested barriers without increasing the code size. With hardware currently available on the CM-5, this scheme can support more than four billion levels of block-nesting in a code, which is more than will ever be required.

C Performance Evaluation

Benchmark results are often reported in terms of Mflops which help summarize the overall performance of the benchmarked machine. However, results in terms of Mflops or benchmark

wall-clock times do not isolate reasons for good or bad performance, nor do they highlight the performance of a machine in specific areas of interest, be it memory access time, cache hit rate, time for a floating point add versus multiply, etc.

In [11] we proposed a methodology to measure detailed machine characteristics with minimal effort (see **Enclosure 7**)

The basic problem in this approach involves measuring delivered execution times of machine instructions by solving a linear system of equations

$$\mathbf{Ax} = \mathbf{t}. \tag{1}$$

where $\mathbf{A} = [a_{ij}]$ $(i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n)$ (preferably $m > n$) is the number of instructions of type j in the benchmark i and t_i is the execution time of the program i. The unknown $\mathbf{x}$ can be solved by using various norm minimization techniques like regression, optimization using linear programming. Prior work by Bard [4, 5] applied regression to estimate the overhead for various services performed by the CP-67 operating system on IBM 360/67. Our work on measuring the performance of Sun SPARCstations matched Bard's findings in most respects. However, they were less than satisfactory. Our experiments and simulations showed that the solution obtained using such norm minimization techniques can be arbitrarily far from the correct solution. This led us to approach the problem of solving Eq. 1 from a different angle [12].

We abstract a benchmark by an $(n-1)$-dimensional hyperplane in $\Re^n$, its execution on the machine being measured by an operating point which lies on this hyperplane. We can estimate the delivered execution time of machine instructions by selecting a representative set of intersection points, these points being defined as the points of intersection of n benchmark hyperplanes. In 1993–94, we formalized our approach and measured the performance of a series of Sun SPARCstations obtaining positive results [12].

During 1994-95, performance vectors for the Cray C90 were computed based on the Perfect suite of benchmarks. These vectors were used to predict application execution time of other applications running on the Cray C90.

Further, we defined a measure called *compliance*. The question is given a set of bench-

marks, how accurately is it possible to compute the performance vectors. This has nothing to do with workload that the benchmarks represent. We assume that the benchmarks accurately represent their respective workloads. The question is given such a set of benchmarks, how accurately can performance predictions be made using those benchmarks. We find that smaller benchmarks like kernels give more accurate predictions than benchmarks composed of applications. This reveals an interesting tradeoff – while application benchmarks best represent the workload on a system, kernel benchmarks are best suited for making performance predictions.

Additional details can be found in Umesh Krishnaswamy's thesis [13].

D Routing Algorithms

Many multistage interconnection networks, such as the Omega Network, the Flip Network, the Baseline Network and the Reverse Baseline Network have been studied over the past 40 years. It turns out that most of these networks are close variants of each other and can be viewed as special cases of the Delta Network [19] [14, page 736] [21].

In 1991-92 we had proposed a generalization of the traditional Delta Network called the *Expanded Delta Network* (EDN) [2], in which each wire is replicated, so that more than one message can travel between adjacent switches simultaneously. EDNs provide multiple paths between any input-output pair, which reduces contention within the network. At the same time, they retain the nice properties of single-path networks, such as digit controlled self routing.

The global router in the MasPar MP-1 and MP-2 [6] is closely related to the EDN. This enabled us to transfer several theoretical results to the MasPar, yielding significant improvements upon their methods.

For instance, MasPar's global router has 3 stages of 64x64 switches. However, the stage 3 switches are not fully exploited. We pointed out that a simple change in the retirement of routing tag digits to the stages allows us to "collapse" stage 3 into stage 2, thus obviating the need for the interconnection between stages 2 and 3 and saving the complexity of 1024 wires across the backplane. We also proved that the pruned-down the router has *exactly* the same

average performance as the original one.

The following sub-sections cover two different kinds of routing algorithms: *on-line* and *off-line*. In an on-line algorithm, network connections are established on the fly by switches in a distributed manner. In an off-line algorithm, network connections are pre-computed (e.g., at compile time) by a central controller; off-line algorithms are useful when the communication pattern to be routed is known at compile-time.

D.1 On-line Routing

An important special case of an EDN is one with only two switch stages. Such a network is easy to build, since it partitions naturally and can be realized on a system with processor boards that communicate through a back plane without active components on the back plane or additional boards to accommodate the network. The global network in the 16K-processor MasPar MP-1 and MP-2 is a 2-stage EDN. The results of this subsection apply to 2-stage EDNs only.

- All deterministic on-line algorithms known so far take $O(\sqrt{N})$ passes in the worst case. Although most permutations take much fewer passes, the worst case permutations tend occur in real-world applications, and make it impossible to predict running times. We have proposed an on-line algorithm whose worst case complexity is only $O(N^{1/4})$ passes. The algorithm chooses two *evil twins* σ and τ, which are network-realizable permutations that are "orthogonal" in some sense. Then the following four steps are repeated until all messages have reached their destinations:

  1. Permute the messages according to σ. There is no contention in the network in this step.
  2. Attempt to send messages to their desired destinations. Unsuccessful messages (due to contention) re-try in subsequent passes.
  3. Permute the messages that remain to be delivered according to τ. There is no contention in the network in this step.
  4. Attempt to send the messages to their desired destinations.

In other words, the algorithm preconditions the routing step with σ and τ alternately.

- The randomized on-line algorithm developed in [3] takes only $O(\log \log N)$ passes with high probability. However, it involves setting each switch randomly and independently, which is not practically feasible. For example, it is not reasonable to expect a 64×64 crossbar to select randomly (and uniformly) one out of 64! possible switch settings within 50–100 nanoseconds.

  This has led us to consider *limited randomization*, in which a switch setting is picked randomly from a smaller set of choices. We show that if each switch can choose only from cyclic shifts, then the above randomized algorithm *still* takes $O(\log \log N)$ passes. In other words, *choosing randomly from cyclic shifts is as general as complete randomization.*

The efficacy of evil-twin and limited randomization techniques is shown in **Enclosures 8,9 and 10**. Further details may be found in Brian Alleyne's thesis [1].

These on-line algorithms are directly adaptable to MasPar's global router. Currently their on-line algorithm takes 43 passes on the average, and more than 250 passes in the worst case (268 passes is the worst that we have discovered so far). In contrast, the use of evil twins or limited randomization techniques described above reduces the average number of passes to from 43 to 29 (a 33% improvement), and reduces the worst case so drastically that it is highly improbable that any permutation will take more than 50 passes.

Some of our on-line techniques are being incorporated in the design of MasPar's next-generation computer.

D.2 Off-line Routing

An off-line algorithm is judged on two counts: the *running time*, which is the time for the algorithm to compute the path for each message; and the *routing time*, which is the time for the messages to move along the paths computed by the algorithm.

- The running time of our algorithm is $\Theta(N \log N)$ sequentially, and $\Theta(\log^2 N)$ on an N-processor PRAM, where N is the number of inputs to the EDN under consideration.

- The routing time of our algorithm is 3 passes, independent of N. This figure reduces to 2 passes in the special case of 2-stage EDNs.

This off-line algorithm is directly applicable to MasPar's global router. We showed that any permutation between the processors can be routed across the global router in exactly 32 passes even in the worst case [20]. MasPar does not provide any off-line algorithm to compare our results against; however, it is worth noting that their on-line algorithm takes 43 passes on the average and more than 250 passes in the worst case.

We coded up the algorithm and it is currently being used by MasPar engineers for in-house applications. The code has also been distributed to several MasPar users all over the US. (A a copy of the code is included as Enclosure 2.)

Enclosure 1 is a letter from the Director of Architecture of MasPar, attesting to the success and desirability of this technology-transfer.

E Dynamic Precision Algorithms

Most computer systems provide the user with a choice of a few standard floating point precisions (*e.g.* 32 bits and 64 bits) and scientists choose the precision that yields an acceptable error in computation. However, it is an overkill to use the same high precision throughout the computation. Dynamic Precision (DP) algorithms vary the precision of the data during program execution in order to improve performance.

We proposed dynamic precision versions of numerical algorithms like Runge-Kutta method and Muller's method [10]. The algorithms were implemented on various supercomputer architectures such as the the MasPar MP-1 and the Cray Y-MP and we observed speed-ups of between 1.5 and 4.

We also investigated architectures suited to implementing DP algorithms. One such architecture is the multigauging architecture, proposed by Snyder [22] and Yang[26]. It has the capability to divide its datapath into several datapaths, of smaller word-length, operating in parallel. Thus, if an algorithm uses a smaller word-length, then it has a greater number of "processors" at its disposal. When the algorithm requires smaller precision, it will run

faster, *not only* because smaller precision arithmetic is intrinsically faster, *but also* because there are more processors available to complete the computation.

For lack of a real multigauging machine, we built a simulator for a multigauging machine that we proposed called P^3. The P^3 is multigauging because it allows disjoint blocks of bits within a word to be manipulated independently and in parallel.

When we ran various DP algorithms on the P^3 simulator, an average speedup of 3 was observed. This work is presented in David Kramer's thesis [9].

F Parallel Job Scheduling

The central problem considered is how to share a parallel machine among several parallel jobs. We call this problem *parallel job scheduling.*

The standard way to run several sequential jobs on a uniprocessor is *time-slicing* or *preemption.* In time-slicing, jobs take turns on the machine, and each time a job gets a turn it resumes from where it left off the previous time.

Time-slicing translates to a parallel setting easily: now parallel jobs take turns using the whole parallel machine. In addition, a fundamentally different possibility opens up, called *space-slicing* or *partitioning.* In space slicing, each job is given a disjoint subset of processors for its own dedicated use. (See **Enclosure 11.**)

Then, between these two extremes of time- and space-slicing, it is possible to formulate countless hybrid policies. To give a very simple example, one might space-slice the machine into partitions, and time-slice each partition among several jobs.

Faced with this gamut of possibilities, it is not at all clear which scheduling policy is best. **Enclosure 12** shows the scheduling policies adopted by several commercial and proposed (paper) systems. The horizontal axis of the table indicates whether or not time-slicing is used; and if so, which variation — gang or local (for now it does not matter what these terms mean). The vertical axis of the table indicates whether or not space-slicing is used; and if so, which variation — static or dynamic. We draw your attention to the fact that all nine cells of the table are occupied by some system or the other, indicating that for each

scheduling policy there is some company or research group that thinks it is a good idea. In short, there is a total lack of consensus on which scheduling policy is best.

It is this problem of finding the best scheduling policy that we address. We demonstrate two main results:

- For static workloads, in which the set of jobs is constant (*i.e.*, all jobs are available at the very beginning, and run for a very long time), the best scheduling policy is pure space-slicing.
- For dynamic workloads, in which the set of jobs varies with time and there is no advance knowledge of when jobs will arrive and how long they will run, the best scheduling policy is pure time-slicing.

This is, of course, a simplified phrasing of results. We have left out several details, such as the model used for jobs and machines, and several other assumptions, simplifications, clarifications, etc. The development of these results in full detail, as well many interesting problems that branch off from this work my be found in Raghu Subramanian's dissertation [23].

G Parallel Virtual Memory

Although well understood in traditional operating systems, virtual memory is not a reality for Massively Parallel computers.

We are experimenting with two classes of paging policies: static and dynamic. In the static case, each PE is responsible for a fixed subset of pages. Whenever anybody faults on a page allocated to PE i, then it is PE i's responsibility to fetch the page from disk. In the dynamic case, each PE is responsible for whatever pages happen to be in its memory at that time — thus ownership of pages changes over time. Each PE services its own page faults. However the changing ownership of a page makes it difficult to track the location of a page when it is needed. (The dynamic page scheme is an extension of Kai Li's virtual shared memory [17].)

Results obtained thus far point to the dynamic policy as a better choice overall.

References

[1] B. D. Alleyne. *Methodologies for Analysis and Design of Data Routers in Large SIMD Computers.* PhD thesis, Dept. of Electrical Engineering, Princeton University, June 1994.

[2] B.D. Alleyne and I.D. Scherson. Expanded delta networks for very large parallel networks. In *Proceedings of the International Conference on Parallel Processing*, pages I–127–131, August 1992.

[3] B.D. Alleyne and I.D. Scherson. Permutation routing in 2-stage recirculating delta networks. In *Symposium on Frontiers of Massively Parallel Computation*, pages 502–503, October 1992. Poster paper.

[4] Y. Bard. Performance criteria and measurement for a time-sharing system. *IBM Systems Journal*, 3:193–216, 1971.

[5] Y. Bard and K. V. Suryanarayana. On the structure of cp-67 overhead. In Walter Freiberger, editor, *Statistical Computer Performance Evaluation*, pages 329–346. Academic Press, Inc., New York, NY, 1972.

[6] T. Blank and R. Tuck. Personal communications, 1991.

[7] Chi-Kai Chien. *Bi-directional Interconnection Networks for Massively Parallel Computers.* PhD thesis, University of California, Irvine, December 1995.

[8] Cray Research, Inc., Eagan, MN. *Cray T3D System Architecture Overview Manual*, 1993.

[9] D.A. Kramer. *Efficient Bit-Parallel Supercomputer Architectures and Algorithms.* PhD thesis, Dept. of Electrical Engineering, Princeton University, January 1993.

[10] D.A. Kramer and I.D. Scherson. Dynamic precision algorithms. In *Symposium on Frontiers of Massively Parallel Computation*, pages 539–540, October 1992. Poster paper.

[11] U. Krishnaswamy and I.D. Scherson. A methodology for performance evaluation of supercomputers. In *Proceedings of the International Parallel Processing Symposium – Parallel Systems Fair*, pages 46–49, April 1993.

[12] U. Krishnaswamy and I.D. Scherson. Measuring beyond flops. Paper in preparation, July 1994.

[13] Umesh Krishnaswamy. *Computer Evaluation Using Performance Vectors.* PhD thesis, University of California, Irvine, December 1995.

[14] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays Trees Hypercubes.* Morgan Kauffman, San Mateo, CA, 1991.

[15] C. Leiserson. Fat trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, C34(10):892–901, October 1985.

[16] C. Leiserson, *et al.* The network architecture of the connection machine CM-5. In *Symposium on Parallel Architectures and Algorithms*, pages 272–285, June 1992.

[17] Kai Li. *Shared Virtual Memory on Loosely-coupled Multiprocessors.* PhD thesis, Yale University, October 1986.

[18] G.J. Lipovski and M. Malek. *Parallel computing.* Wiley & Sons, 1987.

[19] J. H. Patel. Performance of processor-memory interconnections for multiprocessors. *IEEE Transactions on Computers*, C29(10):771–780, October 1981.

[20] I.D. Scherson and R. Subramanian. Efficient off-line routing of permutations on restricted access expanded delta networks. In *Proceedings of International Parallel Processing Symposium*, pages 284–290, April 1993.

[21] H. J. Siegel. *Interconnection Networks for Large-Scale Parallel Processing.* D. C. Heath and Co., Lexington, MA, 1985.

[22] L. Snyder. An inquiry into the benefits of multigauge parallel computation. In *Proceedings of the International Conference on Parallel Processing*, pages 488–492, 1985.

[23] Raghu Subramanian. *A Framework for Parallel Job Scheduling.* PhD thesis, University of California, Irvine, December 1995.

[24] Thinking Machines Corporation, Cambridge, MA. *The Connection Machine CM-5 Technical Summary*, October 1991.

[25] C. L. Wu and T. Y. Feng. On a class of multistage interconnection networks. *IEEE Transactions on Computers*, C29(8):694–702, August 1980.

[26] C. Yang. *An Investigation of Multigauge Architectures.* PhD thesis, Department of Computer Science, University of Washington, 1987.